UNITED STATES PATENT APPLICATION

FOR

MODULARIZED EXTRACTION, TRANSFORMATION, AND LOADING FOR A DATABASE

INVENTOR:

YU GONG

PREPARED BY:

HICKMAN PALERMO TRUONG & BECKER LLP
1600 WILLOW STREET
SAN JOSE, CALIFORNIA 95125
(408) 414-1080

ASSIGNEE:

ORACLE INTERNATIONAL CORPORATION
500 ORACLE PARKWAY
REDWOOD SHORES, CA 94065

"Express Mail" mailing label number EV 323351405 US

Date of Deposit February 20, 2004

MODULARIZED EXTRACTION, TRANSFORMATION, AND LOADING FOR A DATABASE

RELATED APPLICATIONS

[0001] The present application is related to the following applications:

[0002] U.S. Patent Application Serial No. 08/852,968 entitled "Pluggable Tablespaces for Database Systems," filed by William H. Bridge, Jr., Jonathan D. Klein, J. William Lee, Juan R. Loaiza, Alex Tsukerman, Gianfranco Putzolu filed on May 8, 1997 (issued as U.S. Patent No. 5,890,167 on Mar. 30, 1999; referred to as '167), the contents of which are hereby incorporated by reference in its entirety for all purposes; and

[0003] U.S. Patent Application Serial No. 08/865,693 entitled "Tablespace-relative database pointers," filed by William H. Bridge, Jr., Jonathan D. Klein, J. William Lee, Juan R. Loaiza, Alex Tsukerman, Gianfranco Putzolu filed on May 30, 1997, (issued as U.S. Patent No. 6,272,503 on August 7, 2001; referred to as '503), the contents of which are hereby incorporated by reference in its entirety for all purposes.

FIELD OF THE INVENTION

[0004] The present invention relates to database systems. The invention specifically relates to extraction, transformation, and loading for a database.

BACKGROUND OF THE INVENTION

[0005] Many organizations store information critical to their operations in one or more databases. In the complex environments in which these databases operate, it is often necessary to either archive the database for backup and later retrieval or to replicate full or partial databases from one location to another. Both of these operations, replication and

archival/retrieval, involve extraction and loading of data and metadata from a "source" database into a "target" database.

[0006] A first approach for exporting data to a target database, referred to herein as the "database link approach", uses database links. Under this approach, a target database uses a reference or link to a particular set of data in lieu of storing the data itself. The reference allows the data to remain at a source database while treating the data as part of the target database. A problem with this approach is that when a query accesses the linked data as part of the target database, the data must be retrieved from the source database – requiring the data to be transported over a network one piece at a time. The piecemeal transport causes extensive network, time, and processing overhead.

[0007] A second approach for exporting data is the command and data generation approach. In the command and data generation approach, a file of insert commands is generated, one for each record to be exported from the source database. The insert commands conform to a standard database language, such as structure query language (SQL) or any other appropriate format. To import the data, the insert commands are executed at the target database, thereby loading into the target database all of the data from the file of insert commands. A problem with the command and data generation approach is that executing an insert command for each exported record is an extremely slow process. Another problem with the approach is that the export file is subject to file size limitations imposed by the operating system. This means that the command and data generation approach does not work in the case where the aggregate amount of data and metadata exceed a certain predefined limit. A third problem with this approach is that not all types of metadata can be exported from the source database and imported into the target database. In particular, the method cannot handle metadata that can be stored external to the database. Consequently, database

application users must transport data and metadata separately using many different tools. It is normally a tedious and error-prone process.

[0008] Another approach for extraction and loading of database data is the use of transportable tablespaces (TTS): referred to herein as the "TTS approach". A "tablespace" is a collection of storage containers (e.g. data files) used to store data for database objects (e.g. relational tables). In the TTS approach, tablespaces are exported from a source database and imported into a target database. This capability allows the data files of a tablespace to be copied or transported to a database server managing the target database using operating system utilities and allows the data in the data files to be loaded into the database simply by incorporating data files into the set of data files used by the database for storing data. The TTS approach runs much faster than the database link and command and data generation approaches. A problem with the TTS approach is that not all types of metadata associated with the data in the tablespace is transported. In particular, the method cannot extract or load metadata that can be stored external to the database. Consequently, database application users must transport data and metadata separately using many different tools. It is normally a tedious and error-prone process.

[0009] Another approach is the database connection approach. The database connection approach uses processes that extract, transform, and load data from a source database into a target database. The database connection approach works by extracting data row-by-row from the source database, transmitting that data over a network connection from the source database to the target database, transforming the data into a format appropriate for the target database, and loading the data into the target database. One problem with this approach is that it is very slow, primarily because the data for each row is transmitted separately. Another problem with this approach is that not all types of metadata are transported to the

target database. In particular, the method cannot extract or load metadata that can be stored external to the database. This lack of ability to transport certain types of metadata is undesirable for many database applications. Consequently, database application users must transport data and metadata separately using many different tools. It is normally a tedious and error-prone process.

[0010] Therefore there is clearly a need for a method for loading database data into a target database, which allows the efficient and automatic transport of metadata and any associated data.

[0011] The approaches described in this section are approaches that could be pursued, but not necessarily approaches that have been previously conceived or pursued. Therefore, unless otherwise indicated, it should not be assumed that any of the approaches described in this section qualify as prior art merely by virtue of their inclusion in this section.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

[0013] FIG. 1 is a block diagram that depicts an example system for loading database data into a target database according to one embodiment of the invention.

[0014] FIG. 2 is a flow diagram that depicts a process for automatically loading database data into a target database according to one embodiment of the invention.

[0015] FIG. 3 is a flow diagram that depicts a process for automatically extracting database data from a source database according to one embodiment of the invention.

[0016] FIG. 4 is a block diagram that illustrates a computer system upon which an embodiment of the invention may be implemented according to one embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

[0017] A method and apparatus for loading database data into a target database is described. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

GENERAL SUMMARY

[0018] Consider a source database environment and a target database environment each including a database and two applications communicatively coupled to the database: a metadata manipulation application (MMA) and an extraction, transform, and loading (ETL) application. The MMA and ETL applications allow one to add, delete, and modify the data and metadata in the corresponding database. The metadata for each application resides in a location accessible to the MMA and ETL applications, respectively.

[0019] In order to extract database data and metadata from the source database environment and load the data and metadata into the target database environment, the metadata stored in the source database and the source applications is extracted to one or more files. The data associated with the extracted metadata, typically stored in one or more data files at the source database, is made available to the target database along with the extracted metadata. For example, copies of the files containing the data and metadata could be transported via file transfer protocol (FTP) to the target environment. Once the target database server or a process thereto communicatively coupled has access to the files containing the data and metadata, either to the original files or to copies thereof, the target database server analyzes the metadata and incorporates the data in the data files into the

target database by informing the target database server of the location of the data files. The target database server or the process thereto communicatively coupled also incorporates the metadata into the target database and the ETL and MMA applications as appropriate. For example, data from a source database, a metadata description of a table (containing the data) from the source database metadata, and a metadata description of a view of that table from a source MMA are transported via FTP to a target database environment. Once at the target database environment, the metadata is analyzed and the table is added to the target database, the view is added to the target MMA, and the corresponding data is made available to the target database by informing the target database of the location of the data files. Given that the data files are in the target database's native format or one interpretable by the target database server, knowing the location is all that is needed to enable the target database server to access the data from the data files.

[0020] In many cases, at least some of the items to which the metadata correspond will have data associated with them. In some cases, however, the metadata will not have corresponding metadata. In cases where there is no data corresponding to the metadata, the metadata itself is still incorporated into the target database environment and the step of making the data files available to the target database will either not be performed or the data files will be empty.

[0021] In various embodiments described herein, the operations can be performed as a single atomic operation. That is, a full set of data and metadata can be stored at a source environment, transported to a target environment, and incorporated into the target environment as a single atomic operation, allowing the operation to be rolled back if unsuccessful. In various embodiments of the techniques described herein, the transportation of the data is automatic: a user of the system need only provide a specification for the

transportation, which may include names and locations of all the data and metadata; then the system validates the transportation specification, and if the specification is valid, it performs the actual data and metadata transportation without further user intervention.

[0022] Various novel techniques are described herein for extraction and loading of database data and metadata. For the sake of exposition, an example embodiment is described below. Specifics and possible alternatives are given as part of the example embodiment. These specific details in no way limit the present invention, but instead provide illustrations intended to aid understanding.

DEFINITIONS

[0023] View: A view is a logical table defined by a query, where the logical table may be referred to in database commands as a table. The logical table is treated as having the columns returned by the query.

[0024] Sequence: A sequence is a set of non-overlapping identifiers. Each identifier provided based on a sequence is guaranteed to be distinct from ever other identifier provided based on the sequence

[0025] Dimension: A dimensions is a categorization of data in the database table. Each category in a dimension can be subcategorized by another dimension.

[0026] Cube: A cube is a logical organization of data containing multiple dimensions. The "edges" of the cube are defined by contained dimensions. Cells in the cube are definable by a value for each dimension in the cube. Each cell contains values.

[0027] ETL: Extraction, Transformation, and Loading of data from one data location to another.

[0028] ETL Mapping: is a data flow specification detailing which source data impacts which target data and can comprise a reference or link to data that resides outside of the database.

[0029] Database catalog: one of many locations where metadata about database data is stored.

[0030] Transportable tablespace: is a tablespace that can be copied from one database and plugged into another. This is described further in '167.

[0031] Queue: a queue is a storage and dispatching mechanism for applications to communicate that allows applications to communicate in an orderly fashion.

[0032] External table: an external table is a table whose metadata resides in the database and whose data is external to the database.

[0033] Stored procedures: A stored procedure is a program that is stored in the database as metadata. It can be called by database applications.

SYSTEM OVERVIEW

[0034] FIG. 1 is a block diagram that depicts an example system for loading database data into a target database according to one embodiment of the invention.

[0035] The system comprises a network 105 that is communicatively coupled to a source database server 109, a source metadata manipulation application 120, a source extraction transform and loading application 130, a target database server 149, a target metadata manipulation application 160, and a target extraction transform and loading application 170. The network 105 can include a wireless network, dial up access medium, the Internet, a local area network (LAN), or any other appropriate communication network.

[0036] The source database server 109, source MMA 120, source ETL 130 are part of the source database environment 101. The target database server 149, target MMA 160, target ETL application 170 are part of the target database environment 141.

SOURCE DATABASE ENVIRONMENT

[0037] A database server, such as the source database server 109 and the target database server 149, is a combination of a set of integrated software components and an allocation of computational resources, such as memory and processes for executing the set of integrated software components on a processor, where the combination of software and computational resources are used for managing a database. Among other functions of database management, a database server governs and facilitates access to a database, processing requests by database clients to access the database. The database clients of a database server may include other database servers.

[0038] In this example, the source database server 109 manages a single source database 110. Source database 110 and target database 150 are a collection of database objects, the data for which is stored in data files 115 and 155, respectively. The database objects may be any form of structured data. Structured data is data structured according to a metadata description defining the structure. Structured data includes relational tables, object tables, object-relational tables, and bodies of data structured according to the Extensible Markup Language ("XML"), such as XML documents.

[0039] The databases 110, 150 include data stored in data files 115, 155 and database metadata 116, 156, respectively. The database metadata 116, 156 describe core database objects in the databases 110, 150. Herein the term "core database objects" refers to tables, indexes, constraints, or any other appropriate database object whose metadata resides inside of the database so that a database server managing the database can access the database and

maintain the integrity of the database. The term "extended database objects" refers to, for example, a database view, a database sequence, a database dimension, a database cube, an ETL mapping, or a database object, where the metadata for the database object can be stored outside of the source database 110 and the target database 150. The term "database objects" refers to both core database objects and extended database objects.

[0040] The databases 110, 150 are each stored in one or more data files 115, 155. The database metadata 116, 156 define what data files 115, 155 store data for a particular core database object. The data files 115, 155 can be located on a Unix file system, a machine-readable medium, or other appropriate storage mechanism communicatively coupled to the database servers 109, 149. Data files 115 and 155 can also be tablespaces and can be integrated into tablespaces as described in '167 and '503.

[0041] The source database server 109 is communicatively coupled to a source metadata manipulation application 120. A metadata manipulation application, such as the source MMA 120 and the target MMA 160, has its own metadata, 126 and 166, respectively, and is a program for viewing, creating, or changing data in the data files (115, 155, respectively), the database metadata (116, 156, respectively), the MMA metadata (126, 166, respectively), or the ETL metadata (136, 176, respectively). An MMA describes one or more extended database objects, the data for which resides in an associated database (110, 150, respectively).

[0042] Source MMA 120 and target MMA 160 could each be a single process running on one or more processors on a single computer, multiple processes running on one or more processors on a single computer, or multiple processes running on two or more computers. In other examples, there might be no source MMA 120 or target MMA 160 or each MMA could be part of database servers 109, 149, respectively.

[0043] The source database server 109 is also communicatively coupled to a source extraction transformation, and loading application 130. An ETL application, like source ETL application 130 and target ETL application 170, comprises ETL metadata (136, 176, respectively). The source ETL application 130 is a program for extracting, loading, or transforming data associated with the source database 110. The target ETL application 170 is a program for extracting, loading, or transforming data associated with the target database 150. The ETL metadata 136, 176 for each ETL application 130, 170 describes one or more extended database objects. The ETL metadata 136, 176 can be stored outside of the source database 110 and the target database 150. The ETL applications 130,170 can each be a process running on one or more processors on a single computer, multiple processes running on one or more processors on a single computer, or multiple processes running on two or more computers. In other examples, there might be no ETL applications 130, 170 or the ETL applications 130, 170 could be part of the respective database servers 109, 149.

TARGET DATABASE ENVIRONMENT

[0044] The target database server 149 manages the target database 150. The target database server 149 is communicatively coupled to a target metadata manipulation application 160 and a target extraction transformation, and loading application 170. The target MMA 160 and target ETL application 170 are described above.

LOADING DATABASE DATA AND METADATA INTO A TARGET DATABASE ENVIRONMENT

[0045] FIG. 2 is a flow diagram that depicts a process for automatically loading database data into a target database according to one embodiment of the invention. In the example herein, the target ETL application 170 is performing the steps described herein. It is not

critical, however, what application performs the steps. The target MMA 160, the target database server 149, or any other appropriate application could perform the some or all of the steps, and the steps could be performed by a single application or multiple applications.

[0046] First, metadata must be obtained, step 210. In this example, the metadata is stored in one or more files. In other examples, the data could be stored in one or more data structures in memory or in the database itself. The metadata that is transported could include a portion of any or all of the following: source metadata 106, source MMA metadata 126, and source ETL metadata 136. Generally, the metadata is obtained via file transfer protocol (FTP), hypertext transfer protocol (HTTP), secure HTTP (HTTPS), rsync, or over a network using another appropriate protocol. Target ETL application 170 could also read at least a portion of the metadata from a machine-readable medium while the machine-readable medium resides at a location communicatively coupled to the target ETL application 170 and where the metadata was recorded to the machine-readable medium at a location communicatively coupled to the source ETL application 130. A specific example is a target ETL application 170 obtaining metadata from a source ETL application 130 using FTP.

[0047] Once the metadata is obtained, it is analyzed in step 220. In this example, the step 220 of analyzing the metadata includes determining what database objects are represented in the metadata by parsing the file in which the metadata is transported, and recording that information in an internal data structure at the target ETL application 170. The metadata includes one or more definitions of one or more database objects from the source metadata 106, the source MMA metadata 126, and the source ETL metadata 136.

[0048] In one embodiment, as part of obtaining metadata in step 220, an associated extraction process stores source data files 115 on accessible machine-readable medium, at a host location other than the source environment 101 or the target environment 141, or on a

portable machine-readable medium such as compact disks or tapes. In a related embodiment, the machine-readable medium is accessible to the target ETL application 170 and allow the ETL application 170 to have direct access to the extracted source data.

[0049] For example, in the context of FIG. 1, a target ETL application 170 analyzes a file containing:

[0050] source database metadata 116 from a source database 110, where the source database metadata 116 includes core database objects;

[0051] source MMA metadata 126 associated with a source MMA 120, wherein the source MMA metadata 126 describes one or more extended database objects, the data for which is in a source database 110; and

[0052] source ETL metadata 136 associated with a source ETL application 130, wherein the source ETL metadata 136 describes one or more extended database objects, the data for which is in a source database 110.

[0053] Once the metadata is analyzed in step 220, the data associated with the metadata is incorporated in step 230. The metadata being analyzed in step 220 describes one or more database objects the data for which is in a source database 110, and the one or more database objects reside in one or more source data files 115 associated with the source database 110. Herein, the term "source data files" are defined as the original source data files 115 located on the source database 110, or a copy thereof. In some cases, the copy of the source data files 115 are included in or replace the target data files 155. In any case, the copies of the source data files 115, whether or not they are considered target data files 155, are herein termed source data files 115.

[0054] The source data files 115 are in a format that is understandable by the target database 150. In some cases the format of the source data files 115 is native to both the

source database 110 and the target database 150. In other cases, the source data files 115 are in a compatible, but non-native format that is readable, i.e. interpretable, by the target database server 149, even though it is not the target database's 150 native format. The target database server 149 accomplishes this kind of format independence by running a section of code capable of interpreting the data in the non-native format of the source data files 115 in place of the code that is capable of interpreting the data in a format native to the target database 150. In either case (native or non-native format), the data in the source data files 115 is incorporated into a target database 150 by providing the target database server 149 access to the source data files 115 and by signifying location of the source data files 115 to the target database server 149.

[0055] Once the target database server 149 accesses the source data files 115 associated with the metadata analyzed in step 220, the target database server 149 can read and manipulate the data included in the source data files 115.

[0056] As part of incorporating the data in step 230, the target database metadata 156, the target MMA metadata 166, and the target ETL metadata 176 are all augmented as appropriate. For example, if data associated with a database sequence is loaded into the target database 150, and the metadata for that database sequence was from the source MMA metadata 126, then the metadata for that database sequence is incorporated into the target MMA metadata 166. Incorporation of metadata into the target metadata 156, target MMA metadata 166, and target ETL metadata 176 can be performed using any appropriate interface. For example, the target database server 149 might have a special interpreter to incorporate the metadata directly from the file containing the source database metadata 116. The target MMA 166 and target ETL application 170 could have application program interfaces (APIs) for adding the metadata.

[0057] In one embodiment, as part of incorporating the data in step 230, the target ETL application 170, invokes the target database's transportable tablespace mechanism to "plug in" the data and thus create the tablespace in the target database 150. As the result of creating the tablespaces, all tables' data and metadata appear in target database 150. However, at this point, the extended metadata has not been created in the target database environment 141.

[0058] In one embodiment, if no data is associated with the source database metadata 116, source MMA metadata 126, or source ETL metadata 136, then no data associated with the metadata is incorporated as part of step 230.

[0059] The example described in FIG. 2 enables analysis and incorporation of metadata and data into a target database 150, and thereby, enables efficient loading of data and complex metadata from source database 110, source MMA 120, and source ETL application 130.

EXTRACTING DATABASE DATA AND METADATA FROM A SOURCE DATABASE ENVIRONMENT

[0060] FIG. 3 is a flow diagram that depicts a process for automatically extracting database data from a source database according to one embodiment of the invention.

[0061] First, the metadata is extracted, step 310. In this example, a human operator signifies what metadata to extract via a graphical user interface to the source ETL application 130. The source ETL application then uses an API to the source database server 109 and the source MMA 120 to extract the appropriate data and metadata. In other examples, any operator, human or automated, specifies to any process communicatively coupled to the source database server 109, including the source MMA 120, to extract the data and metadata as described herein. The extracted metadata describes one or more database objects, the data for which resides in one or more source data files 115 associated with said source database

110. The metadata that is transported could include metadata from any or all of the following: source metadata 106, source MMA metadata 126, and source ETL metadata 136. In this example, the extracted metadata is stored in one or more files. In other examples, the data could be extracted and stored in one or more data structures in memory or in a database.

[0062] In one embodiment, as part of step 310, the metadata are analyzed for dependencies in order ensure that no dangling references exist. For example, if a view is present in the extracted metadata, but the tables are required to define the view are not included in the module, the view is removed from the metadata. In one embodiment, the dependencies in the metadata are also analyzed to ensure proper order of loading into the target database.

[0063] In one embodiment, as part of step 310, meta-metadata is created to represent the transportable tablespace that contains data (i.e., a set of tablespaces and metadata). This meta-metadata is created by the source ETL application 130 and is stored in memory data structures on the source database server 109. In a related embodiment, this meta-metadata is saved into one or more files and later imported into target ETL application 170. In a related embodiment, source and target ETL applications, when loaded with the meta-metadata for the transportable module are able to load the data associated with the meta-metadata.

[0064] In step 320, the source data files 115 are made accessible to the target database server 149. The files can be made available via FTP, HTTP, HTTPS, rsync, or over a network using another appropriate protocol. Making the data available could also be accomplished by causing a machine-readable medium to bear the data, e.g. writing the data to a compact disk (CD), and making that machine-readable medium accessible to the target database server 149. For example, a source ETL app 130 extracts source data files 115 from a source database 110 and makes the data files available to a target ETL application 170 via

FTP. The target ETL application 170 subsequently makes the data files available to a target database server 149. In one embodiment, if no data is associated with the source database metadata 116, source MMA metadata 126, or source ETL metadata 136, then target database server 149 are not provided access to any source data files 115. In another embodiment, if no data is associated with the source database metadata 116, then target database server 149 are is provided access to empty source data files 115.

[0065] In various embodiments, as part of step 320, metadata from source metadata 116, source MMA metadata 126, or source ETL metadata 136, are saved in one or more files. In a related embodiment, the one or more files contain the metadata of objects such as views, dimensions, sequences, etc. In a related embodiment, the file is prepared for transportation to target machine and for later loading into target metadata 156, target MMA metadata 166, or target ETL metadata 176.

[0066] In step 330, access to the metadata extracted in step 310 is made available to the target ETL application 170. In this example, the target ETL application 170 analyzes and inputs the metadata. In other examples, a process communicatively coupled to the target database server 149 analyzes and inputs the metadata, and, in such examples, the metadata would be made available to that process.

[0067] As noted above, the metadata can be stored either in a file or in memory. If the metadata is stored in a file, access to it can be provided by transporting it to a location accessible to the target ETL application 170 via FTP, HTTP, HTTPS, rsync, or over a network using any appropriate protocol. It is also possible to cause a machine-readable medium to bear the data, e.g. writing the data to a CD, and making that machine-readable media accessible to the target ETL application 170. If the metadata is stored in memory, then it can be made available to the target ETL application in any appropriate manner including

remote procedure calls and Transmission Control Protocol / Internet Protocol (TCP/ IP) sockets.

[0068] For example, a source ETL application 130 provides a target ETL application 170 access to metadata extracted in step 310 by transporting, via FTP, a copy of the extracted metadata to location accessible to the target ETL application 170.

[0069] The example depicted in FIG. 3 enables extraction and delivery of data and metadata from a source database 110 to a target database 150, and extraction and delivery of metadata from applications in a source database environment 101 to those in a target database environment 141. The extraction of data and metadata enabled by the example in FIG. 3 combined with the loading enabled by the example in FIG. 2 provide an efficient method for extraction of data and metadata for core and extended database objects from a source database environment 101 and loading of that data and metadata into the target database environment 141 not possible previously.

HARDWARE OVERVIEW

[0070] FIG. 4 is a block diagram that illustrates a computer system 400 upon which an embodiment of the invention may be implemented according to one embodiment of the invention. Computer system 400 includes a bus 402 or other communication mechanism for communicating information, and a processor 404 coupled with bus 402 for processing information. Computer system 400 also includes a main memory 406, such as a random access memory (RAM) or other dynamic storage device, coupled to bus 402 for storing information and instructions to be executed by processor 404. Main memory 406 also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor 404. Computer system 400 further includes a read only memory (ROM) 408 or other static storage device coupled to bus 402 for storing static

information and instructions for processor 404. A storage device 410, such as a magnetic disk or optical disk, is provided and coupled to bus 402 for storing information and instructions.

[0071] Computer system 400 may be coupled via bus 402 to a display 412, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device 414, including alphanumeric and other keys, is coupled to bus 402 for communicating information and command selections to processor 404. Another type of user input device is cursor control 416, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 404 and for controlling cursor movement on display 412. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane.

[0072] The invention is related to the use of computer system 400 for implementing the techniques described herein. According to one embodiment of the invention, those techniques are performed by computer system 400 in response to processor 404 executing one or more sequences of one or more instructions contained in main memory 406. Such instructions may be read into main memory 406 from another machine-readable medium, such as storage device 410. Execution of the sequences of instructions contained in main memory 406 causes processor 404 to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

[0073] The term "machine-readable medium" as used herein refers to any medium that participates in providing data that causes a machine to operation in a specific fashion. In an

embodiment implemented using computer system 400, various machine-readable media are involved, for example, in providing instructions to processor 404 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 410. Volatile media includes dynamic memory, such as main memory 406. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 402. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

[0074] Common forms of machine-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punchcards, papertape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

[0075] Various forms of machine-readable media may be involved in carrying one or more sequences of one or more instructions to processor 404 for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system 400 can receive the data on the telephone line and use an infra-red transmitter to convert the data to an infra-red signal. An infra-red detector can receive the data carried in the infra-red signal and appropriate circuitry can place the data on bus 402. Bus 402 carries the data to main memory 406, from which processor 404 retrieves and executes the instructions. The instructions

received by main memory 406 may optionally be stored on storage device 410 either before or after execution by processor 404.

[0076] Computer system 400 also includes a communication interface 418 coupled to bus 402. Communication interface 418 provides a two-way data communication coupling to a network link 420 that is connected to a local network 422. For example, communication interface 418 may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 418 may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface 418 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

[0077] Network link 420 typically provides data communication through one or more networks to other data devices. For example, network link 420 may provide a connection through local network 422 to a host computer 424 or to data equipment operated by an Internet Service Provider (ISP) 426. ISP 426 in turn provides data communication services through the worldwide packet data communication network now commonly referred to as the "Internet" 428. Local network 422 and Internet 428 both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 420 and through communication interface 418, which carry the digital data to and from computer system 400, are exemplary forms of carrier waves transporting the information.

[0078] Computer system 400 can send messages and receive data, including program code, through the network(s), network link 420 and communication interface 418. In the

Internet example, a server 430 might transmit a requested code for an application program through Internet 428, ISP 426, local network 422 and communication interface 418.

[0079] The received code may be executed by processor 404 as it is received, and/or stored in storage device 410, or other non-volatile storage for later execution. In this manner, computer system 400 may obtain application code in the form of a carrier wave.

EXTENSIONS AND ALTERNATIVES

[0080] In the foregoing specification, embodiments of the invention have been described with reference to numerous specific details that may vary from implementation to implementation. Thus, the sole and exclusive indicator of what is the invention, and is intended by the applicants to be the invention, is the set of claims that issue from this application, in the specific form in which such claims issue, including any subsequent correction. Any definitions expressly set forth herein for terms contained in such claims shall govern the meaning of such terms as used in the claims. Hence, no limitation, element, property, feature, advantage or attribute that is not expressly recited in a claim should limit the scope of such claim in any way. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.